第三届“图灵杯”NEUQ-ACM 程序设计大赛（个人赛）

题解

好学的 coco

原题再现

coco 是一个好学的男生,一天 coco,田鼠和男神 zcx 一起去图书馆约自习,看到图书馆进进出出的人,coco 突然想知道图书馆里面最少能容纳有多少人(设一开始图书馆人数为零),于是问了田鼠和男神 zcx,男神 zcx 想了一想,自豪的说出了自己的想法,没过多久 coco 也想出来了.这时候田鼠满脸迷茫,看着男神 zcx 和船长都装逼成功了,田鼠想了想还是没想出来(毕竟田鼠),田鼠心想不能认怂啊,于是田鼠想请让在坐的聪明的你帮他解决这个问题.

已知每个学生的图书证都有一个特殊的编号n(1<=n<=10^6),+ n表示编号为n的学生进入图书馆,- n表示编号为n的学生离开图书馆.

输入

第一行为k表示已知的进出图书馆的人数,接下来的k行为图书馆进出人的记录.

输出

图书馆最少有多少人

样例输入

```
6
+ 12001
- 12001
- 1
- 1200
+ 1
+ 7
```

样例输出

```
3
```

解题思路

因为每个学生图书证编号 n,1<=n<=10^6,所以可以通过一个长度为 10^6+1 的数组 book[10^6+1]纪录每个学生是否在图书馆内,如果学生 i 在图书馆则 book[i]为 1,不在的话为 0,模拟学生进出图书馆情况,纪录图书馆内人员最大值就可以了,注意如果有一个在 k 组纪录里,没进去就出去的人,说明他之前在图书馆

按上面过程模拟就可以了

参考答案

此标程为单组输入输出,若测试数据为多组请自行修改。

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include<iostream>
#include<string>
#include<map>
#include<set>
#include<algorithm>
#include<queue>
#include<vector>
#include<time.h>
```

```cpp
#include<assert.h>
using namespace std;
int book[1000010];
int main(){
    int n;scanf("%d",&n);
    string t;
    int k;
    int in=0;
    int ans=0;//the max value library can hold
    for (int i=1; i<=n; i++) {
        cin>>t;
        scanf("%d",&k);
        if (t=="+") {
            book[k]=1;
            in++;
            if (in>ans) {
                ans++;
            }
        }
        else{
            if (!book[k]) {
                ans++;
            }
            else{
                in--;
                book[k]=0;
            }
        }
    }
    printf("%d\n",ans);
    return 0;
}
```

Zcx 学数学

原题再现

zcx 大神是一个数学狂,他对周期函数很感兴趣,于是他提出了一个类似周期 函数的问题,假设一个人从原点(0,0)出发一直按照某种路径走路,问你他能否到达一个点,设这个点为(a,b),(1<=a,b& lt;=10^9),你只需要判断能否到达这个点,如果能到达就输出 Yes,不能到达的话输出 No.我们用一个字符串 s 表达这个路径,其中 L 表示向左移 动,R 表示向右移动,U 表示向上移动,D 表示向下移动

输入

输入多组输入,每组包含a,b,s其中a,b为终点坐标,s为字符串,表示周期路径

输出

Yes或No

样例输入

```
2 2
RU
1 2
RU
-1 1000000000
LRRLU
0 0
D
```

样例输出

```
Yes
No
Yes
Yes
```

解题思路

由于终点(a,b),1<=a,b<=10^9,时间限制为 1s,普通的完全模拟是会超时的,但是注意这个人的行走方式是具有周期性的,我们只需要纪录每个周期他的坐标变化量 dx,dy,以及他在第一周期内行走的每个位置的坐标(xi,yi),如果存在正整数 k,使得 xi+k*dx=a,yi+k*dy=b,就可以了,难点在于如何处理 dx,dy 为 0 的情况

参考答案

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include<iostream>
#include<string>
#include<map>
#include<set>
#include<algorithm>
#include<queue>
#include<vector>
#include<time.h>
```

```
#include<assert.h>
typedef long long  ll;
const int inf=1e+9;
using namespace std;
struct node{
    int x;
    int y;
}point[110];
int q[110],p[110];
int a,b;
int main(){
    scanf("%d%d",&a,&b);
    int ok=0;
    int x0=0,y0=0;
    char s[110];scanf("%s",s);
    int len=strlen(s);
    point[0].x=0;
    point[0].y=0;
    for (int k=0; k<len; k++) {
        if (s[k]=='U') {
            y0++;
            point[k+1].y=point[k].y+1;
            point[k+1].x=point[k].x;
        }
        else if(s[k]=='D'){
            y0--;
            point[k+1].y=point[k].y-1;
            point[k+1].x=point[k].x;
        }
        else if(s[k]=='L'){
            x0--;
            point[k+1].x=point[k].x-1;
            point[k+1].y=point[k].y;
        }
        else if(s[k]=='R'){
            x0++;
            point[k+1].x=point[k].x+1;
            point[k+1].y=point[k].y;
        }
    }
    for (int i=0; i<len; i++) {
        int dx=a-point[i].x;
        int dy=b-point[i].y;
        if (x0 && y0 && dx%x0==0 && dy%y0==0 && dx/x0>=0 &&
```

```
dx/x0==dy/y0) {
            ok=1;
            break;
        }
        if (!x0 && !y0 && !dx && !dy) {
            ok=1;
            break;
        }
        if (!x0 && point[i].x==a && y0 && dy%y0==0 && dy/y0>0) {
            ok=1;
            break;
        }
        if (!y0 && x0 && point[i].y==b && dx%x0==0 && dx/x0>0) {
            ok=1;
            break;
        }
        if (x0 && y0 && dx%x0==0 && dy%y0==0 && dx/x0>0 &&
dx/x0==dy/y0) {
            ok=1;
            break;
        }
    }
    if (ok || (a==0 && b==0)) {
        printf("Yes\n");
    }
    else
        printf("No\n");
    return 0;
}
```

橙子姐姐的数列

原题再现

橙子姐姐有各项都为正数的数列a1,a2,a3,⋯,a(n*t),对任意i>n,满足ai=a(i-n),聪明的橙子姐姐可以求出最长非递减数列的长度吗?明显不能。请你帮他计算结果

输入

第一行包括两个数n,t(1<=n<=100,1<=t<=10^7),第二行为a1,a2,⋯,an.

输出

最长非递减数列的长度

样例输入

```
4 3
3 1 4 2
```

样例输出

```
5
```

解题思路

当 t 很小的时候很容易处理,我们只需要考虑当 t(t>>n)很大的时候
一个长度为 n 的数列,最多有 n-1 个非递减对,例如 1 2 3 4,(1,2) (2,3) (3,4).
.那么我们需要考虑 a1,a2,⋯,a(n^2)的最长非递减序列的长度 len 和 a1,a2,⋯,an 里面出现次数最多的数字的出现次数 num,答案其实就是 num*(t-n)+len

参考答案

此标程为单组测试。如果题目为多组输入输出，请自行修改。

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include<iostream>
#include<string>
#include<map>
#include<set>
#include<algorithm>
#include<queue>
#include<vector>
#include<time.h>
#include<assert.h>
typedef  long  long   ll;
const  int  inf=1e+9;
using  namespace  std;
int  a[11000];
int  num[310];
int  b[11000];
int  dp[310];
int  main(){
        int  n,t,k;
        int  max=-1;
```

```
    cin>>n>>t;
    k=min(n,t);
    for  (int  i=1;  i<=n;  i++)  {
        cin>>a[i];
        num[a[i]]++;
        max=std::max(max,num[a[i]]);
    }
    for  (int  i=2;  i<=k;  i++)  {
        for  (int  j=1;  j<=n;  j++)  {
            a[(i-1)*n+j]=a[j];
        }
    }
    int  ans=-1;
    for  (int  i=1;  i<=n*k;  i++)  {
        dp[a[i]]++;
        for  (int  j=0;  j<=a[i]-1;  j++)  {
            dp[a[i]]=std::max(dp[a[i]],dp[j]+1);
        }
        ans=std::max(ans,dp[a[i]]);
    }
    ans+=max*(t-k);
    cout<<ans<<endl;
    return  0;
}
```

田鼠 PK 船长

原题再现

船长和田鼠最近在玩一个取石子的游戏，石子共有 n 堆，两人轮流取石子，每次船长先取，取得石子数只能是质数或一，而且只能取完一堆后才能取下一堆，下一次取得堆编号必须必上一个小，无法再取的人将会输掉比赛。假如两人都是用最优策略，请问谁会赢得比赛呢？

输入

第一行一个整数 T ，代表一共有 T 组测试数据，接下来一行输入n，下一行是 n 个整数代表每一堆的石子数

输出

如果船长胜利输出"yes",田鼠胜利输出"no"，均不带引号。

样例输入

```
2
18467 6334
1
19169
5
11478 29358 26962 24464 5705
```

样例输出

```
yes
yes
yes
```

解题思路

对于一堆而言，所有 4 的倍数只能变为不是 4 的倍数，所有不是 4 的倍数都可以变为 4 的倍数。所以先手是 4 的倍数必输，不是 4 的倍数必胜。对于多堆而言，只要有一个先手必胜的堆那么从这个堆开始拿就是先手必胜。

参考答案

```
#include <stdio.h>
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    int n,t;
    bool ans;
//  freopen("in.txt","r",stdin);
    scanf("%d",&t);
    ofstream outfile("out.txt",ios::out);
    while(t--)
    {
        ans=0;
        scanf("%d",&n);
        while(n--)
        {
```

```
                int m;
                scanf("%d",&m);
                if(m%4)ans=1;
            }
            if(ans)printf("yes\n");
            else printf("no\n");
//          if(ans)outfile<<"yes\n";
//          else outfile<<"no\n";
        }
        return 0;
}
```

田鼠买酒喝

原题再现

田鼠上次偷喝酒被抓住后，船长好好的惩罚了他一顿，现在田鼠安分了许多。
有一天，田鼠实在是犯了酒瘾，只好带着好朋友 LJC 来找船长：“老板，要不你卖我们酒喝吧！？”船长皱了皱眉头：“我这酒可是无价的，不卖!”田鼠哇的一下哭了出来，赖在甲板上不肯走，船长无奈了，说：“这样吧，我这里有一堆的石子，我先拿一堆，你再拿一堆，最后 LJC 拿一堆，如果这三堆石子可以组成勾股数的话，那我就把酒给你们了!”田鼠傻了：“什么是勾股数？!”LJC 说：“勾股数就是有三个数字，其中存在两个数字的平方和等于第三个数的平方。”田鼠恍然大悟，但是还是不会算，你能帮帮他吗？

输入

多组输入，每组输入有三个正整数 A,B,C 分别是船长的石子数，田鼠的石子数，LJC 的石子数。(三个数随机排列)

输出

对于每组输入，如果田鼠能拿到酒则输出“YES”，否则输出“NO”，每组输出占一行，具体看样例。

样例输入

```
3 4 5
1 2 3
```

样例输出

```
YES
NO
```

解题思路

排一下序在判断 A^2+B^2==C^2 ，数据非有序

参考答案

```
#include<cstdio>
#include<cstdlib>
#include<math.h>
#include<iostream>
#include<cstring>
#include<map>
#include<set>
#include<list>
#include<stack>
#include<algorithm>
#include<queue>
#include<vector>
#include<time.h>
#include<fstream>
using namespace std;
const int INF=1<<30;
typedef long long ll;

int a[5];
```

```
int main(){
	freopen("in.txt","r",stdin);
	char filename[]="out.txt";
	ofstream ofs;
	ofs.open(filename);
	while(~scanf("%d%d%d",&a[0],&a[1],&a[2])){
		sort(a,a+3);
		if(a[0]*a[0]+a[1]*a[1]==a[2]*a[2]){
			//printf("YES\n");
			ofs<<"YES"<<endl;
		}
		else //printf("NO\n");
			ofs<<"NO"<<endl;
	}
	return 0;
}
```

田鼠看热闹（i）

原题再现

田鼠闲来无事到集市上偷吃干货，突然被人群吸引住了，他挤进人群中一看，原来他们在玩一个叫做“开心翻转”的游戏，游戏初始如下：

有一串只含 0 和 1 的数字序列（例如：1 0 1 0 1 1 0 ），长度为 n，玩家每次可以选择一个位置进行操作，每次操作可以将指定位置以及指定位置两边的数字翻转（如果两边有数字的话），每次翻转（0 变成 1,1 变成 0）需要消耗一个游戏币，如果可以经过 K 次操作全部翻转成 1 的话，大喊一声“为了部落!”，如果无论怎么翻转都无法得出答案的话大喊“荣誉属于联盟!”，有解得情况下，消耗最少游戏币的玩家获得胜利，无解的情况下，最快喊出答案的玩家获得胜利。

小田鼠看着奖品直流口水，所以想请你帮帮他得出答案。

输入

多组数据输入，每一组输入第一行一个数字 n(1<=n<=10^6)

接下来的一行输入长度为 n 的数字序列（只含 0,1）

输出

对于每组输入，如果有解，则输出最少消耗的游戏币，如果不可解，则输出“-1”具体看样例。

样例输入

```
3
0 0 0
2
1 0
```

样例输出

```
1
-1
```

样例解释

第一个样例只需要翻转第二个数字即可

第二个样例无论怎么翻转都无法得到全是 1 的情况

解题思路

从最左边开始判断，因为翻转为连续三个翻转，所以一边记录当前这个位置被翻转过几次，如果奇数则变化了，如果次数为偶数说明没有变化过，然后不断根据当前位置是否是 0 来判断要不要翻转下一个位置（选择翻转的地方左边和右边都被翻转）扫一次就可以，最后判断末尾位置有没办法被翻转为 1（或者是否已经为 1）

参考答案

```
#include<cstdio>
#include<cstdlib>
#include<math.h>
#include<iostream>
#include<cstring>
#include<map>
#include<set>
#include<list>
#include<stack>
```

```
#include<algorithm>
#include<queue>
#include<vector>
#include<time.h>
#include<fstream>
using namespace std;
const int INF=1<<30;
typedef long long ll;

int n,m;
int a[2000000];
int c[2000000];
int b[2000000];
int main(){
	int i,j;
	freopen("in.txt","r",stdin);
	char filename[]="out.txt";
	ofstream ofs;
	ofs.open(filename);
	while(scanf("%d",&n)!=EOF){
		for(i=1;i<=n;i++)scanf("%d",&a[i]);
		if(n==1){
			if(a[1]==0)ofs<<1<<endl;
			else ofs<<0<<endl;
			continue;
		}

		//·-²»·-µÚÒ»¸ö
		if(a[1]==0){
			int ans1=1,ans2=1;
			memset(c,0,sizeof(c));
			c[1]++;
			c[2]++;
			for(i=3;i<=n;i++){
				if((c[i-1]+a[i-1])%2==0){
					ans1++;
					c[i]++;
					c[i+1]++;
				}
			}
			if((a[n]+c[n])%2==0)ans1=INF;

			memset(c,0,sizeof(c));
			c[1]++;
```

```
                c[2]++;
                c[3]++;
                for(i=3;i<=n;i++){
                    if((c[i-1]+a[i-1])%2==0){
                        ans2++;
                        c[i]++;
                        c[i+1]++;
                    }
                }
                if((a[n]+c[n])%2==0)ans2=INF;

                int ans=min(ans1,ans2);
                if(ans==INF)ofs<<-1<<endl;
                else ofs<<ans<<endl;
            }
            else{
                int ans=0;
                memset(c,0,sizeof(c));
                for(i=3;i<=n;i++){
                    if((c[i-1]+a[i-1])%2==0){
                        ans++;
                        c[i]++;
                        c[i+1]++;
                    }
                }
                if((a[n]+c[n])%2==0)ans=-1;
                ofs<<ans<<endl;
            }
        }
        return 0;
}
```

田鼠看热闹（ii）

原题再现

田鼠在集市中继续游荡着，突然又被人群吸引住了，他挤进人群中一看，原来他们在玩一个叫做“幸运小子”的游戏，游戏初始如下

1. 游戏有 n 个玩家，编号从 1-n，并且这 n 个玩家围成一个环形
2. 由主持人随机说一个数字 m
3. 每次从玩家 1 顺时针开始数 m 个数（包括玩家 1），数到第 m 个人的编号是 k，则 k 玩家出局，由此时在玩家 k 下一位的玩家继续开始下一轮游戏，直到剩下一个人为止，则这个人可以拿走奖品

田鼠一看，我去，这不就是约瑟夫问题么？赶紧找到正确的位置加入，开心的拿到了奖品，，可是这时候主持人大喊：“桥豆麻袋！（稍等一下）这位先生，你要拿走奖品还需要回答一个问题！”田鼠心虚了：“什么问题！？”主持人说：“请问，刚才的出局顺序是什么！”田鼠懵逼了，只好请你来救救他。

输入

多组数据输入，每一组输入数据包含两个数字 n(1<=n<=10^6)，m(1<=m<=10^9)

输出

对于每组输入，先输出一个正整数，表示田鼠一开始加入的编号，并且这个编号是最后的存留编号，之后换行，然后输出n-1个数，顺序输出出队的玩家的编号，两个数之间用空格隔开，具体看样例

样例输入

```
7 3
```

样例输出

```
4
3 6 2 7 5 1
```

样例解释

当 n=7,m=3 的时候，游戏从玩家 1 开始，出局顺序为 3->6->2->7->5->1 最后编号 4 胜利，这也是田鼠初始站的位置

解题思路

本题是约瑟夫问题的变种，区别于原来约瑟夫问题只求最后一个人，这次要求的是所有出队的顺序，正常模拟情况下一定会超时，根据原来约瑟夫问题DP的做法，我们想到了用递推的方式来逐步求解，当当前人数为n时，起始位置知道，用m取模可得下一位的位置是左数第k个，问题是怎么得到下一位的位置呢？我们使用线段树来进行优化，用线段树维护区间人数，然后使用二分搜索查找前k个人的位置，再使用一个L,R数组每次维护每个人相邻的人的号数，O（1）的时间找到去掉这个人后的下一个人的号数，整体时间复杂度大概是O(nlognlogn)，标程里用的是0-n-1，所以答案输出时候+1

参考答案

```
#include<cstdio>
#include<cstdlib>
#include<math.h>
#include<iostream>
#include<cstring>
#include<map>
#include<set>
```

```
#include<list>
#include<stack>
#include<algorithm>
#include<queue>
#include<vector>
#include<time.h>
#include<fstream>
using namespace std;
const int INF=1<<30;
typedef long long ll;

const int MAXN=1200000;
int tri[MAXN*4],ans[MAXN*4],markL[MAXN],markR[MAXN],biao[MAXN];
int n,m,cnt;

void build(int l,int r,int k){
	if(l==r){
		tri[k]=1;
		biao[l]=k;
		return ;
	}
	int mid=(l+r)/2;
	build(l,mid,k*2+1);
	build(mid+1,r,k*2+2);
	tri[k]=tri[k*2+1]+tri[k*2+2];
}

void update(int k){
	while(k){
		k=(k-1)/2;
		tri[k]=tri[k*2+1]+tri[k*2+2];
	}
}

int query(int l,int r,int L,int R,int k){
	if(l==L&&r==R){
		return tri[k];
	}
	int mid=(L+R)/2;
	if(l>mid){
		return query(l,r,mid+1,R,k*2+2);
	}
	else if(r<=mid){
		return query(l,r,L,mid,k*2+1);
```

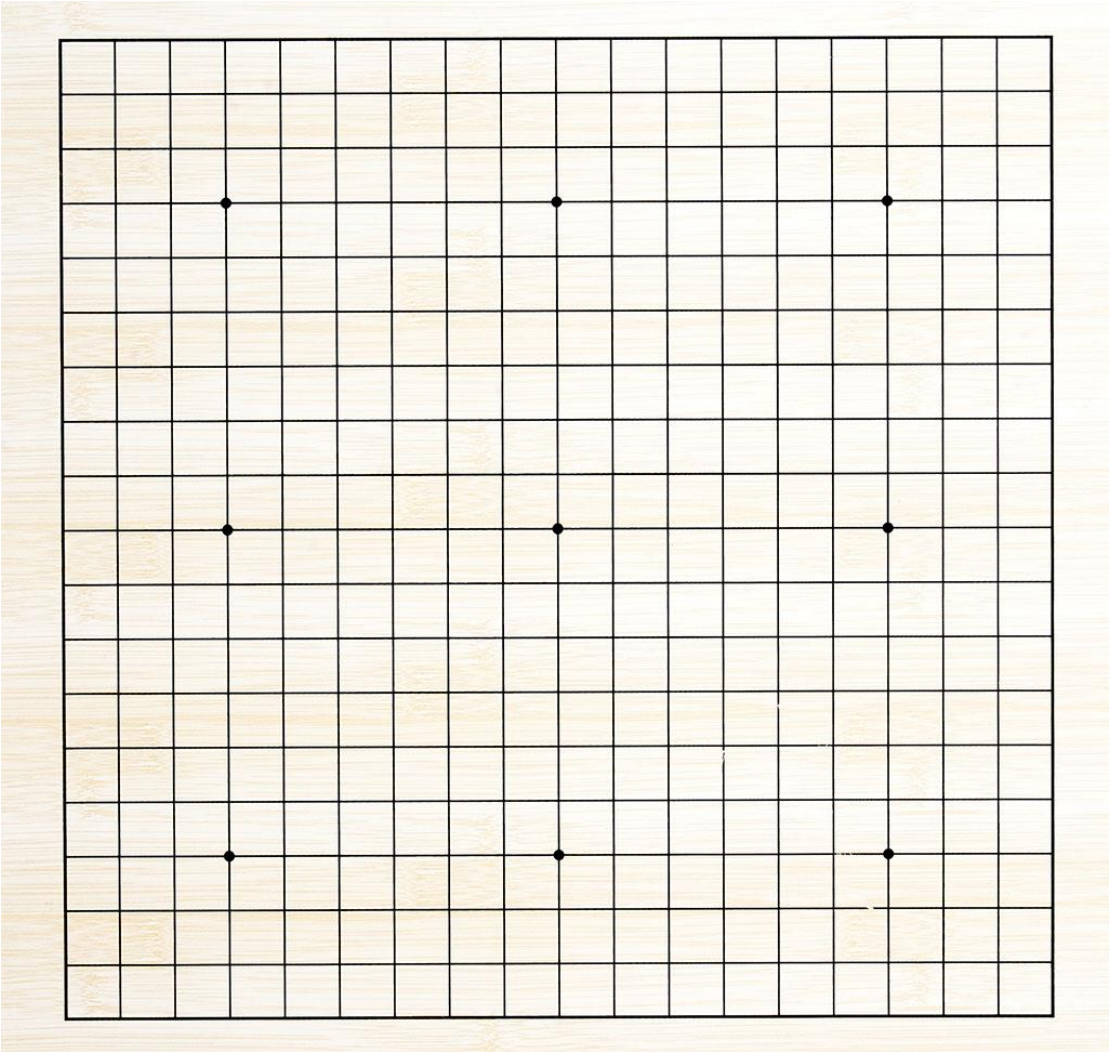
```
        }
        else{
                return query(l,mid,L,mid,k*2+1)+query(mid+1,r,mid+1,R,k*2+2);
        }
}

int Search(int x){
        int i,j;
        int l=0,r=n-1;
        while(true){
                //cout<<"dd";
                int mid=(l+r)/2;
                int c=query(0,mid,0,n-1,0);
                if(c>=x){
                        r=mid;
                }
                else{
                        l=mid;
                }
                if(r-l==1){
                        int c=query(0,l,0,n-1,0);
                        if(c==x)return l;
                        else return r;
                }
        }
        return l;
}

void init(){
        memset(tri,0,sizeof(tri));
        memset(markL,0,sizeof(markL));
        memset(markR,0,sizeof(markR));
        int i,j;
        for(i=0;i<n-1;i++){
                markL[i]=i+1;
        }
        markL[n-1]=0;
        for(i=n-1;i>0;i--){
                markR[i]=i-1;
        }
        markR[0]=n-1;
}

void GetQuery(){
```

```
    int i,j,t;
    int s=0;
    while(cnt<(n-1)){
        //cout<<"cc";
        t=m%(n-cnt);
        if(t==0)t=(n-cnt);
        int c=query(0,s,0,n-1,0);
        int w=(c+t-1)%(n-cnt);
        if(w==0)w=n-cnt;
        //cout<<w<<endl;
        ans[cnt]=Search(w);
        tri[biao[ans[cnt]]]=0;
        update(biao[ans[cnt]]);
        markL[markR[ans[cnt]]]=markL[ans[cnt]];
        markR[markL[ans[cnt]]]=markR[ans[cnt]];
        s=markL[ans[cnt]];
        //cout<<s+1<<endl;
        cnt++;
    }
    ans[cnt++]=s;
    //cout<<query(0,2,0,n-1,0)<<"      "<<query(0,7,0,n-1,0)<<endl;
    //cout<<Search(1)<<"      "<<Search(5)<<endl;
}

int main(){
    freopen("in.txt","r",stdin);

    int i,j,ct=1;
    while(cin>>n>>m){
        char filename[]="out1.txt";
        filename[3]=ct++;
        ofstream ofs;
        ofs.open(filename);
        cnt=0;
        init();
        build(0,n-1,0);
        GetQuery();
        ofs<<ans[cnt-1]+1<<endl;
        for(i=0;i<cnt-1;i++){
            ofs<<ans[i]+1;
            ofs<<(i!=cnt-2?' ':'\n');
        }
    }
    return 0;}
```

橙子姐姐的围棋

原题再现

橙子姐姐是东秦有名的围棋高手当他打败东秦所有围棋高手后，独自对着棋盘发呆。他发现，他下的奇葩围棋棋盘由 m*n（1<m,n<10000）个点组成了若干个正方形（每个小格子都是 1 个正方形），现在橙子姐想知道这个棋盘中有多少个正方形。图为 19*19 的棋盘



输入

第一行输入 t(1<=t<=15)表示有 t 组输入数据,接下来的 t 行每组占 1 行每行 2 个数 m 和 n

输出

输出正方形个数

样例输入

```
19 19
233 666
```

样例输出

```
2109
15892464
```

解题思路

考察以某个点为右下角的正方形个数位 min(i,j)，所以将各个点作为正方形右下角时的正方

形个数相加就是所有正方形数量

参考答案

```
#include<iostream>
#include<cstdio>
#include<algorithm>
using  namespace  std;
long  long  n;
long  long  m;
long  long  co;
int  main()
{
        int  t;
        cin>>t;
        while(t--)
        {
                cin>>m>>n;
                co=0;
                for(long  long  i=1;i<m;i++)
                {
                        for(long  long  j=1;j<n;j++)
                        {
                                co=(co+min(i,j));
                        }
                }
                cout<<co<<endl;
        }
                return 0;
}
```

```
//#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
int n;
int m;
int co;
int main()
{
    while(~scanf("%d%d",&m,&n))
    {
        co=0;
        for(int i=1;i<m;i++)
        {
            for(int j=1;j<n;j++)
```

```
                {
                        co=(co+min(i,j));
                }
        }
        printf("%d\n",co);
    }
    return 0;
}
//
```

橙子姐姐的梦

原题再现

橙子姐姐偷走了船长的宝藏并买下了 n（1<n<=10^6）个城市，编号从 1 到 n，第 i 个城市的价格为 a_i，(0<a_i<=10^5,且为整数)，但是这 n 个城市之间没有路，不是互相连通的，因此他请来了工程师 zcx 来帮他修路。橙子姐姐想尽量多修路，但黑心的 zcx 为了赚更多的钱，告诉橙子姐姐 1 个城市最多只能有 1 条路连接其他城市，并且修一条路的费用等于橙子姐姐购买这条路相连的两个城市的费用总和。精明的橙子姐姐由于经费原因只能修 k（1<=k<=10^3,k<n/2）条路，请帮橙子姐姐计算一下他最少花多少经费

输入

第一行输入一个整数 t（1<=t<=3）表示有 t 组测试数据，对于每组数据，第一行输入 2 个数 n，k 分别代表城市数量 n 和道路数量 k，第二行输入 n 个数分别代表城市的价格 a_1，a_2，a_3…….a_n。

输出

对于每组数据，输出橙子姐姐总共花的钱

样例输入

```
2
5 1
3 2 1 5 4
6 2
2 5 9 7 1 1
```

样例输出

```
3
9
```

解题思路

题意就是求前2*k个最小数的和，排序求和是nlogn会超时，建立大小为2*k的大顶堆，扫描数组，更新堆，复杂度是nlonk，当然还有更快的算法

参考答案

```
#include <cstdio>
#include<iostream>
#include<algorithm>
#define LEFT(i) 2*(i)+1
#define RIGHT(i) 2*(i+1)
using namespace std;
const int maxn=1000010;
int a[maxn];
void max_heapify(int *arr,int index,int len)//建立大顶堆的过程，求前 k 个最小，要健最大堆，调整堆
{
    int l=LEFT(index);//所有操作类似于堆排序
    int r=RIGHT(index);
    int largest;
    if(l<len && arr[l]>arr[index])
        largest=l;
```

```
    else
        largest=index;
    if(r<len && arr[r]>arr[largest])
        largest=r;
    if(largest != index)
    {//将最大元素提升，并递归
        swap(arr[largest],arr[index]);
        max_heapify(arr,largest,len);//递归
    }
}

void build_maxheap(int *arr,int len)//开始建立大顶堆是必须的
{
    int i;
    if(arr==NULL || len<=1)
        return;
    for(i=len/2+1;i>=0;--i)
        max_heapify(arr,i,len);
}

void k_min(int *arr,int len,int k)
{
    int i;
    build_maxheap(arr,k);
    for (i = k;i < len;i++)
    {
        if (arr[i] < arr[0])//就是这一个地方跟堆排序不一样，这里只是交换比堆顶大的元素。
        {
            arr[0] = arr[i];
            max_heapify(arr,0,k);
        }
    }
}
int main()
{
    int n,k;
    //freopen("data.txt","r",stdin);
    int t;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d%d",&n,&k);
          k*=2;
    for(int i=0;i<n;i++)
```

```
        scanf("%d",&a[i]);
    k_min(a,n,k);
    long long co=0;
    for(int i=0;i<k;i++)
        co+=a[i];
    cout<<co<<endl;
    }
    return 0；
}
```

橙子姐姐的商业帝国

原题再现

橙子姐姐是个无良商人，拥有一个吃货公司，船长是这家公司的老顾客。这家公司有 n 个工厂（n<=10^5）编号为 1，2，…,n，分别负责制造不同的食品（比如船长喜欢喝的酒，杯面，薯片……）。其中一个工厂负责把各个零食打包成大礼包，然后卖给船长。现在对于这 n 个工厂有 n-1 个传送带（传送带可以有 2 个方向运货物），任意两个工厂之间有且仅有 1 条传送带。受传送带的速度限制和承重限制，1 分钟只能往单一方向运送 1 单位的食品。现在给你各个工厂已经生产好的食品数量，请你帮橙子姐姐计算一下需要多少单位时间能把各个工厂现有的全部零食运送到组装的工厂。

输入

第一行 1 个整数 T，表示数据组数。
每组数据第一行有 2 个数据 n，s，代表工厂数量和组装工厂的编号
接下来的 n-1 行，每行 2 个数 i,j 表示工厂 i 和 j 之间有一条传送带。
接下来 n 行每行一个数 x（ x<100000），表示各工厂的部件数量

输出

对每组数据输出 Y，表示把所有部件送到组装厂的最短时间

样例输入

```
2
3 2
1 2
2 3
1
2
1
5 1
1 2
2 3
2 4
2 5
5
1
2
2
1
```

样例输出

```
1
6
```

解题思路

把组装的工厂定为根节点，对于根节点的每个子树的时间分别是 t1,t2,t3…..总时间为 max（t1,t2,t3…）
把与根距离相同的分为一层，传输时考虑传送带可能有空闲的时候，子树传输所以货物的时间为货物量+空闲时间。算出每一层的排队延迟，再与没有货物的曾作比较，就可以算出最后的空闲时间了。Dfs 来搜索层数

参考答案

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<cstring>
using namespace std;
const int maxn=100010;
int first[maxn];
int d[maxn],num[maxn];
int deep;
struct node
{
	int v;
	int next;
}edge[maxn*2];
void dfs(int u,int ba,int deepth)
{
	if(num[u]) deep=max(deep,deepth);
	d[deepth]+=num[u];
	for(int i=first[u];i!=-1;i=edge[i].next)
	{
		int v=edge[i].v;
		if(v==ba) continue;
		dfs(v,u,deepth+1);
	}
}
int main()
{
	//freopen("data2.txt","r",stdin);
	int t;
	scanf("%d",&t);
	for(int cas=1;cas<=t;cas++)
	{
		int n,root;
		int co=0;
		memset(first,-1,sizeof(first));
		memset(d,0x3f,sizeof(d));
		scanf("%d%d",&n,&root);
		for(int i=1;i<n;i++)
		{
			int v,u;
			scanf("%d%d",&u,&v);
			edge[co].next = first[u];
```

```
            edge[co].v = v;
            first[u] = co++;
            edge[co].next = first[v];
            edge[co].v = u;
            first[v] = co++;
        }
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&num[i]);
        }
        long long ans=0;
        for(int i=first[root];i!=-1;i=edge[i].next)
        {
            int v=edge[i].v;
            deep=0;
            memset(d,0,sizeof(d));
            dfs(v,root,0);
            long long t1=0,t2=0;
            for(int i=0;i<=deep;i++)
            {
                t1+=d[i];
                if(d[i]==0&&t1+t2<=i)
                    t2++;
            }
            ans=max(t1+t2,ans);
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

ACM Club